Final Report

TNW 99-01

Budget 61-7994

VIDEO IMAGE PROCESSING

TO CREATE A SPEED SENSOR

by
Daniel J. Dailey
and
L. Li

Department of Electrical Engineering
University of Washington
Seattle, Washington, 98195

Transportation Northwest (TransNow)
University of Washington, Box 352700
Seattle, WA 98195-2700

November 1999

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the data presented herein. This document is disseminated through the Transportation Northwest (TransNow) Regional Center under the sponsorship of the U.S. Department of Transportation UTC Grant Program and through the Washington State Department of Transportation. The U.S. government assumes no liability for the contents or use thereof. Sponsorship for the local match portion of this research project was provided by the Washington State Department of Transportation. The contents do not necessarily reflect the official views or policies of the U.S. Department of Transportation, the Washington State Department of Transportation, or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

Abstract

Image processing has been applied to traffic analysis in recent years, with different goals. In this report, a new approach is presented for extracting vehicular speed information, given a sequence of real-time traffic images. We extract moving edges and process the resulting edge information to obtain quantitative geometric measurements of vehicles. This differs from existing approaches because we use simple geometric relations obtained directly from the image instead of using reference objects to perform camera calibrations. Our method allows the recovery of the physical descriptions of traffic scenes without explicit camera calibration. In this report, extensive experiments using images from active TMS (Transportation Management System) freeway cameras are reported. The results presented in this report demonstrate the validity of our approach which requires neither direct camera control nor placement of a calibration object in the environment. We further argue that it is straightforward to extend our method to other related traffic applications

TABLE OF CONTENTS

.

1 Introduction

In recent years, image processing has been applied to the field of traffic research with goals that include queue detection, incident detection, vehicle classification, and vehicle counting [1, 2, 3, 4, 5, 6, 7].

This report explicitly recognizes that speed is an important parameter in traffic analysis. Relatively few efforts have attempted to measure speed by using video images from uncalibrated cameras. Some preliminary research on pixel speed estimation in images appears in Soh et al [6] and Picton [8]. A review of the literature on physical speed estimation shows that almost all of the algorithms involve some man-made reference information. For example, Worrall et al [9] describes an interactive tool for performing camera calibration. In this interactive application, an operator first identifies vanishing points from parallel road marks and then places a rectangular grid on the image for calibration. Dickinson and Waterfall [10] and Ashworth et al [11] make speed measurements from the known physical distance between two detection windows placed on the road image. Similarly, several other papers [12, 13] suggest estimating speed by first placing two detection lines (separated by a known distance) and then measuring travel times between the lines. Houkes [14] determined four reference points to form a rectangle before taking the off-line measurements. In that case, the camera had to remain in the same position during all measurements for the process to be valid.

In this research, we assumed that we had no control over camera movements and thus could not directly obtain information such as camera focus, tilt, or angle. We further assumed that the camera parameters could change any time without our knowledge. In our project, we were monitoring congested freeways and had neither the ability nor the authority to set permanent marks on the road.

We believe on-line calibration is a necessary step in using the large, installed base of TMS cameras. We propose that exact calibration is not necessary to estimate speed using our algorithm; instead, we use information inherently available in the image. We focus on a 1-D geometry for the traffic on the road. Using a car length distribution from our previous research [15], we propose a novel method that extracts scaling signatures and computes the speed distribution on the basis of the geometric relationships in the image.

1.1 ASSUMPTIONS

To clarify the problem presented here, we made several assumptions in our work:

a) Finite speed [16]: The speed of a vehicle has both physical and legal limits.

b) Movement is smooth [16]: No sudden changes of directions are expected between frame intervals (330ms).

c) Motion is constrained to the road plane [17], and thus we are posing camera calibration as a 1-D geometry problem.

In this work, we used 320x240 gray-scale images at a frame rate of 3 frames per second (fps). These are demonstrated to be adequate for reliable analysis, as well as being small enough to allow efficient processing.

1.2 REPORT OVERVIEW

Our algorithm for speed extraction first applies a series of operators to single images to create a set of enhanced images. We then use this set of enhanced images to create a speed estimation algorithm. In this report, we first describe the single image operations and then present the overall algorithm as applied to a group of images. Chapter 2 introduces major procedures for each single image. These consist of a preprocessing step, a moving-edge detection step, a morphological operation step, and a convex hull and bounding box extraction

step. In Chapter 3, we discuss geometric relation analysis, as well as distance and speed estimation algorithms for an image sequence. Chapter 4 presents the field trials, experimental results, and discussion. In Chapter 5, we present conclusions about the effectiveness of the algorithm.

2 Single Frame Processing

The single image processing steps are shown in Figure 2.1. These steps include preprocessing, moving edge detection, morphological operations, and convex hull and bounding box extraction. The next section describes details of the pre-processing under the assumptions presented earlier.

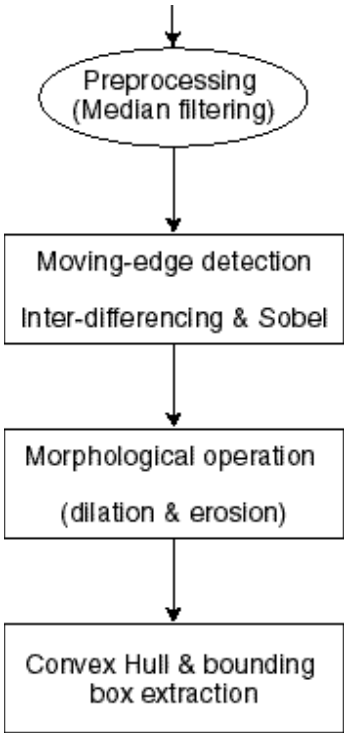


Figure 2.1 Image processing flow for a single image

2.1 PREPROCESSING

The traffic images have a noise component from several interference sources. The types of noise include the following [18]:

1) Salt-and-pepper noise, which occurs when an image is coded and transmitted over a noisy channel or degraded by electrical sensor noise, as in video cameras.
2) Convolutional noise (blurring), which produces images that are degraded by lens mis-focus, motion, or atmospheric turbulence, such as adverse weather conditions.

Both noise sources contribute to high-frequency noise components. In our process, median filtering is used to reduce this high-frequency noise. It preserves the edge information required by our algorithm. Edges are a key image feature, as they remain prominent despite the variations in the traffic scene's ambient lighting. Our median filter uses a 3x3 kernel to remove high frequency noise from the image.

The 3x3 kernel moves row by row, pixel by pixel. A pixel is regarded as the center of the 3x3 window. The median value of the set of nine pixels in the 3x3 neighborhood is used as the new filtered value of the pixel at the center. This way, impulse noise with extreme values can be suppressed. Since the total area in our process is fixed in advance, locating the median value is fast. It is the fifth value in the sorted array [19].

The next section describes the moving edge detection module of our algorithm. The algorithm uses the images preprocessed by the median filtering just presented.

2.2 MOVING-EDGE DETECTION

Moving edge detection is applied to extract the moving parts from a complex background in an image sequence. The static background is then deleted to locate the moving objects.

2.2.1 Sobel Edge Detector

Let $I(i,j)$ denote the pixel value being processed. Its neighbors are considered to determine whether it is on an edge or not. Usually a 3x3 or 5x5 neighbor window is used for one pixel. In our work, a 3x3 window is used for processing, as shown in the matrix below. For the pixel $I(i,j)$, the eight neighbors are I_1 through I_8.

I_3	I_2	I_1
I_4	$I(i,j)$	I_8
I_5	I_6	I_7

The Sobel edge magnitude is computed is [20] as

$$I_s(i, j) = (u^2 + v^2)^{1/2}, \quad (1)$$

where

$$u = (I_5 + 2I_6 + I_7) - (I_1 + 2I_2 + I_3), \quad (2)$$

and

$$v = (2I_8 + I_1 + I_7) - (I_3 + 2I_4 + I_5). \quad (3)$$

The gradient is computed as

$$G_s(i, j) = \tan^{-1}(u / v). \quad (4)$$

The above computational process moves a 3x3 window with the current pixel as the window center. After the magnitude is obtained, a threshold can be used to determine which pixel is on an edge. If the Sobel magnitude is below the threshold, the pixel will be discarded. This means that the magnitude response is not strong enough to claim an edge point. The selection of an appropriate threshold is dependent on the content of the images.

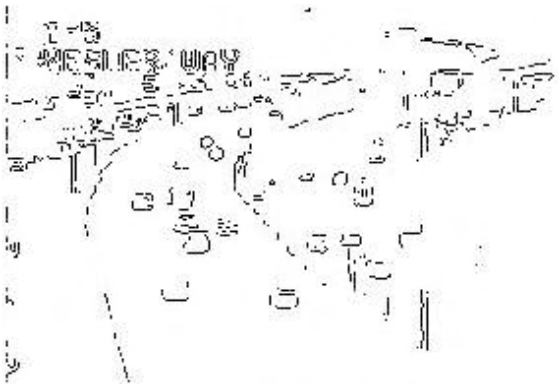
An example is shown in Figure 2.2. We assert that most of the edges are detected by Sobel edge detection.

2.2.2 Moving Edge Detection

In previous work by Gil [21], moving objects were segmented from the traffic background with a motion detection algorithm based on a multi-resolution relaxation. This resulted in a set of coarse binary masks for each vehicle. A refinement process was then applied to obtain a more accurate description. In multi-resolution relaxation, both the starting and ending resolution need to be selected on the basis of engineering judgment. Fathy and Siyal [2] present a window-based edge detection method that combines morphological edge detectors and a

median filtering. However, their process [2] requires the user to pre-place all the detection windows at the key regions across the lanes, and therefore the user needs to have detailed knowledge of the road. Kudo [22] applies a one-dimensional gradient operation to a sub-region with a window along the road, which falls into the same category.

Original image



Sobel edge image

Figure 2.2 Sobel edge detection

In this process, we use image differencing to extract motion information. There are two basic differencing methods in the literature: 1) background differencing and 2) interframe differencing. In background differencing, a reference frame that contains no moving vehicles is subtracted from each frame. In real world applications, where the ambient lighting varies rapidly, the reference frame needs to be updated regularly to reflect the current background and

to provide reliable segmentation. This reference frame can be obtained by either grabbing a frame when no vehicles are presented or by multi-frame accumulation [23]. Several methods are suggested by Fathy and Siyal [2] and Koller et all [17] to update the background image. However, these methods are slow and computationally expensive and thus cannot meet real-time processing requirements. Furthermore, on congested freeways (the domain of interest) it is difficult to obtain images with no vehicles that match the present light level. Therefore, to mitigate these problems, we adopt the inter-frame differencing method to eliminate the complex background and detect the moving vehicles.

Cai [23] used forward and backward image differencing and then extracted common regions corresponding to the moving areas. Instead of extracting regions, Vieren [24] proposed a method to combine inter-frame differencing and a differential operator to extract moving edges. In our process, we combine inter-frame differencing with the Sobel edge detector to extract the moving edges. To emphasize the movement signature, we use three sequential images and process each image relative to its previous and subsequent images. In this way, we separate the movement from the static background.

Our algorithm is applied to three images: the previous temporal image (I_p), the current image of interest (I_c), and the next temporal image (I_n):

$$Edge\_image = Sobel(I_p - I_c) \otimes Sobel(I_n - I_c) \quad . \tag{5}$$

That is:

1) Take the difference between the previous image I_p and the current image I_c.
2) Take the difference between the next image I_n and the current image I_c.

3) Sobel edge operators are applied to these two different images to get two edge images.
4) Compare the magnitudes of all edge pixels in the two edge images resulting from the Sobel edge operator with a magnitude threshold. If the magnitude of a pixel is less than the threshold, then it is set to be 0. Otherwise, it is set to be 1. This produces two binary edge images.
5) Create the intersection of the two binary edge images. Extract common moving edges present in the original current image, I_c.

This process produces an edge image for the current image of interest from which we will extract individual vehicle information in the next chapter.

Figure 2.3 A typical sequence

Example images for the above process are shown in Figures 2.3, 2.4, 2.5, and 2.6. Figure 2.3 shows three original successive frames in an image sequence. Figure 2.4 shows the edge

image of the difference image between the first two frames. Figure 2.5 shows the edge image of the difference image between the second and third frames. Figure 2.6 shows the final moving edge image. Almost all of the moving edges are extracted successfully.

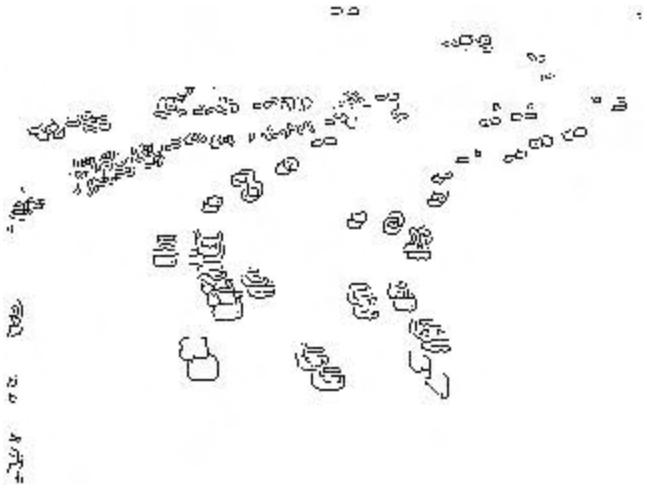


Figure 2.4 Sobel edges in the difference image between the first two frames

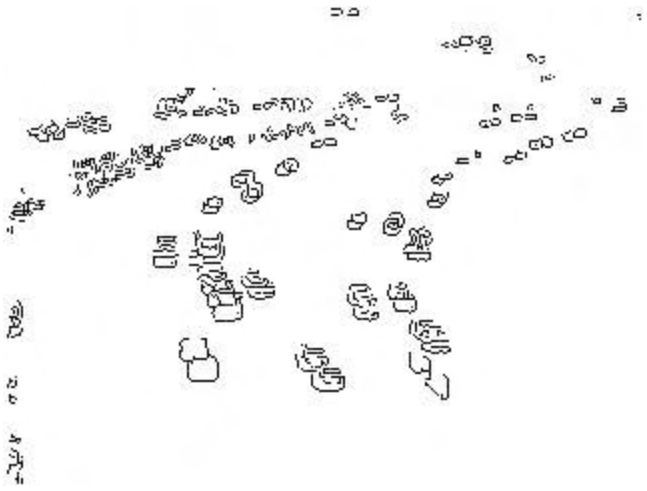


Figure 2.5 Sobel edges in the difference image between the second and third frames

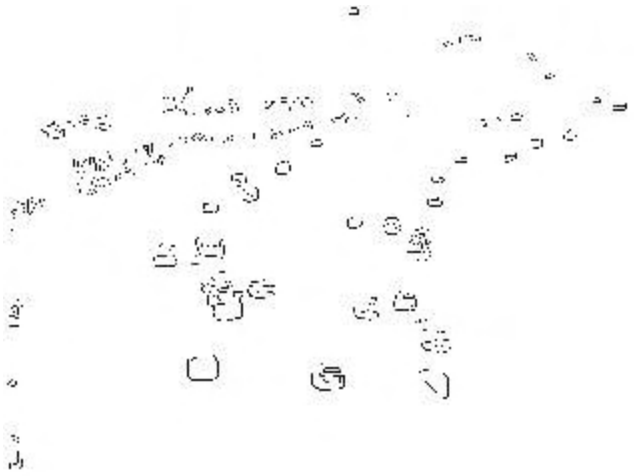


Figure 2.6 Moving edges

2.3 MORPHOLOGICAL OPERATION TO OBTAIN MOVING BLOBS

In the moving edge image just described, there are always gaps along the edges. To obtain a profile of the vehicle, we need to enhance the moving edges. This enhancement uses the morphological operators dilation and erosion with an appropriate structural element. The result of sequentially applying dilation and erosion [25] is to remove specific image features smaller than the structural element without affecting the large features of interest.

Dilation and erosion are two basic morphological operations, which will be discussed first. Dilating an object is to translate all its points with regard to a structural element followed by union operation. On the other hand, eroding an object is to translate all its points first by using a structural element and then to conduct the intersection operation to get the final result. This way, dilation expands an object and erosion shrinks it by the size of the specified structural element.

Images are dilated with the max operation. They are eroded with the min operation. A structural element of N by N is used to define the *max* or *min* operation regions. To process an image pixel, the region containing the pixel of interest and its (N-1) by (N-1) neighboring pixels is processed, and the maximum or the minimum value is obtained.

That is, for an image matrix with *m* by *n*, and a given element size *N*,

```
1) sort the pixel values in the N by N neighborhood
2) put the maximum in the dilation image matrix
3) put the minimum in the erosion image matrix.
```

Since we are processing binary images, the above sorting process can be simplified to test whether the pixel value is 1 or 0. In this way, the operations are very fast.

The selection of the size of the structural element *N* depends on the range of the size of the vehicles of interest and the range of distance between the vehicles. For example, a vehicle smaller than the structural element in the image will be removed by erosion. On the other hand, several vehicles close to each other will be merged to a single blob after dilation. While our goal is to extract as many vehicles as possible, we do not require every vehicle in the image to be identified for our algorithm to accurately estimate speed. Using morphological operations, some vehicles will be removed by erosion because of their small size in the image (these have been deemed too small to be useful in speed estimation) or will be merged by dilation because of their proximity (these have been deemed inappropriate because of possible occlusion effects).

Figures 2.7 and 2.8 show some morphological examples. Figure 2.7 shows the result of dilation operation. Different structural element sizes (3x3 and 4x4) are used. Observe that dilation with larger sized structural elements will make some blobs merge together. Figure 2.8 shows erosion operations. Erosion with larger sized structural elements causes some vehicles to disappear from the image.

Original Binary image

After dilation by a 3x3 structural element

After dilation by a 4x4 structural element

Figure 2.7 Dilation examples

(1) original binary image

(2) After erosion by a 3x3 structural element

(3) After erosion by a 4x4 structural element

Figure 2.8 Erosion examples

Figure 2.8 Erosion examples The result of sequentially applying dilation and erosion [25] is to remove specific image features smaller than the structural element without affecting the large features of interest. An example is shown in Figure 2.9, where an image is first dilated and then eroded by the same 3x3 structural element.

Original binary image

After dilation

After erosion

Figure 2.9 Dilation followed by erosion

At this stage, the image of interest has been enhanced to emphasize the moving vehicles that appear as blobs in the resulting image.

2.4 VEHICLE PROFILE APPROXIMATION

2.4.1 Convex Hull Extraction

After the application of morphological operators above, moving edges are filled and appear as solid moving "blobs." To characterize the blobs, we use a convex hull to approximate the contour of the vehicles. In many cases, a convex hull is a good approximation of the projection of a car [21, 17].

In the image produced by the procedure in the previous section, the background is full of 0s, and only points inside and along the contours of the blob are of value 1. We select the contour points by searching each scanline to find the rightmost or leftmost end of a blob.

Not all contour points selected by this method will belong to the convex hull. Therefore, we need to select those points that are actually on the hull. Koller [17] proposed a convex hull extraction method that is suitable for our purpose. The procedure is to define a convex hull point, $P_2(x_2,y_2)$, by its location related to its preceding point, $P_1(x_1,y_1)$, and following point $P_3(x_3,y_3)$. A threshold T is used to determine the associated orientation of these three points, where

$$T \dagger \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}. \qquad (6)$$

A positive value of T indicates that those three points are in counter-clockwise order along the contour. A negative T value indicates that they are in clock-wise order.

This algorithm is used to efficiently obtain all the points on the convex hull, as shown in the following description:

1) If contour point P_2 is on the left side of the contour, T is computed to check whether it is positive (counter-clockwise). If so, P_2 is

regarded as being on the convex hull. If not, P_2 does not belong to the hull.

2) Similarly, if P_2 is on the right side of the contour, T is computed to check its sign. Now, contrary to the above left-side case, if T is positive, it means P_2 is not on the convex hull. If T is negative (clockwise), P_2 belongs to the hull and should be retained.

Figure 2.10 shows an example where a convex hull is extracted by using the above method.

Moving blobs for convex hull extraction

Convex Hull for one blob

Figure 2.10 Convex hull extraction example

2.4.2 Bounding Box Extraction

To obtain scaling information directly from the image rather than using explicit camera calibration, we exploit the known geometric relationships in the images. We do this by constructing a bounding box to enclose the convex hull. This bounding box is used to isolate the area of interest in the image and is similar to window (or key region) processing described by Fathy and Siyal [26] and Stewart et al [27]. However, unlike window processing, we are only interested in looking for some simple geometric relations inside the box.

To obtain the bounding box from the convex hull with vertices x_i and y_i, the algorithm, as seen in Figure 2.11, is:

1) Arrange all the x_i values as an array and find the minimum and maximum.
2) Do the same for all y_i values.
3) The resulting box is the rectangle with vertices (counter-clockwise order):
4) (min_x_i, min_y_i)(max_x_i, min_y_i)(max_x_i, max_y_i)
5) (min_x_i, max_y_i).

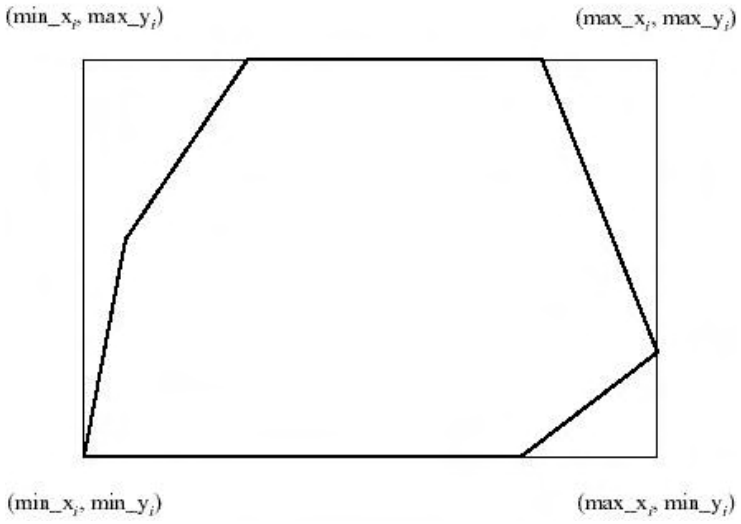
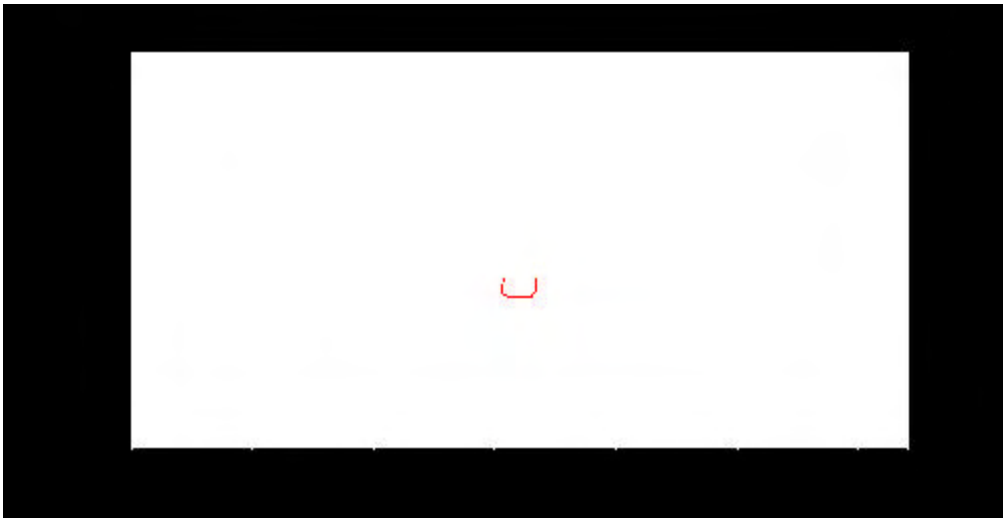


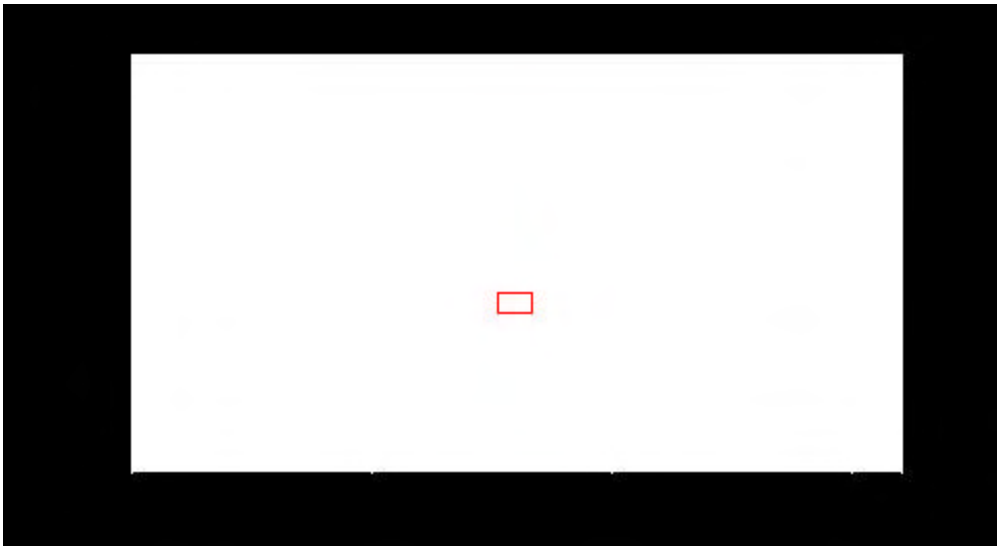
Figure 2.11 Bounding box enclosing a convex hull

This procedure is applied to each image in an image sequence, and we obtain a series of convex hulls and bounding boxes that will be used to estimate the real travel distance and speed, as covered in the next chapter.

Figure 2.12 shows an example of extracting the bounding box from a convex hull.



Convex Hull for bounding box extraction



Bounding box extracted

Figure 2.12 Bounding box extraction

3 Geometric Analysis and Speed Estimation From an Image Sequence

The result from Chapter 2 is a series of convex hulls and bounding boxes. This chapter describes utilizing the geometric features and this series of hulls and blobs for distance and speed computation.

To estimate speed, we first obtain the direction of motion of each vehicle and then compute the best fit line through the centroids of the convex hulls found in a series of images and associated with a single vehicle. A threshold on the correlation coefficient [28] for the centroids is used as the colinearity criterion to identify a single vehicle track. The best fit line for the direction of travel is used to obtain the pixel length of the vehicle, and we exploit a simple triangular relationship in the bounding box to get the pixel length of the vehicle, which is then used to compute the scale information in the images. Ground truth distance is estimated by using scale information along the direction of motion, and these distances, with the frame rate of the video sequence, are used to estimate speed.

3.1 DIRECTION OF MOTION: a

We assume that the vehicles make no sudden changes in directions between successive video frames. This assumption allows us to track individual vehicles through successive frames. We identify a single vehicle track by requiring that the centroids of the convex hull be colinear in successive frames, as shown in Figure 3.1. The linear regression correlation coefficient r for least square straight line fitting, as presented by Bevington [28], is the criterion for determining the colinearity of centroids.

From experiments, we claim that we are able to identify a single vehicle in a succession of images if the colinearity of the centroids produces a linear regression correlation coefficient r greater than 0.90, where

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}, \tag{7}$$

and x_i and y_i are the coordinates of convex hull centroids.

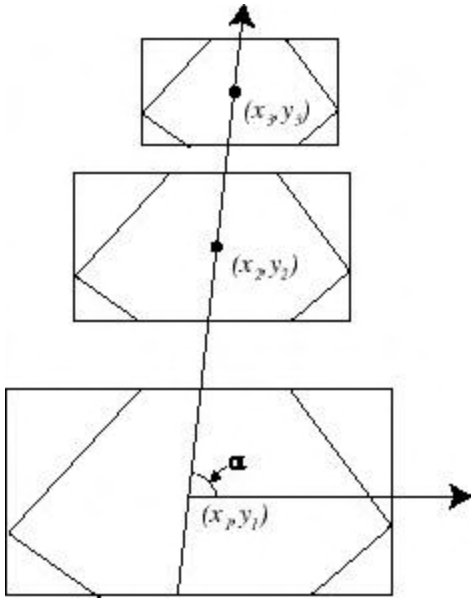


Figure 3.1 Colinearity of convex hull centroids

3.2 GEOMETRIC RELATION INSIDE THE BOUNDING BOX

To get the scale information from the images, we exploit the triangular relationship within the bounding box, as shown in Figure 3.2. The pixel length, *L_pixel*, of a vehicle is estimated along the best fit line (*L*) indicating the direction of travel. It is estimated to be the length of the cord along the best fit line that intersects the bounding box.

$$L\_pixel = \frac{box\_width}{\sin\theta}. \tag{8}$$

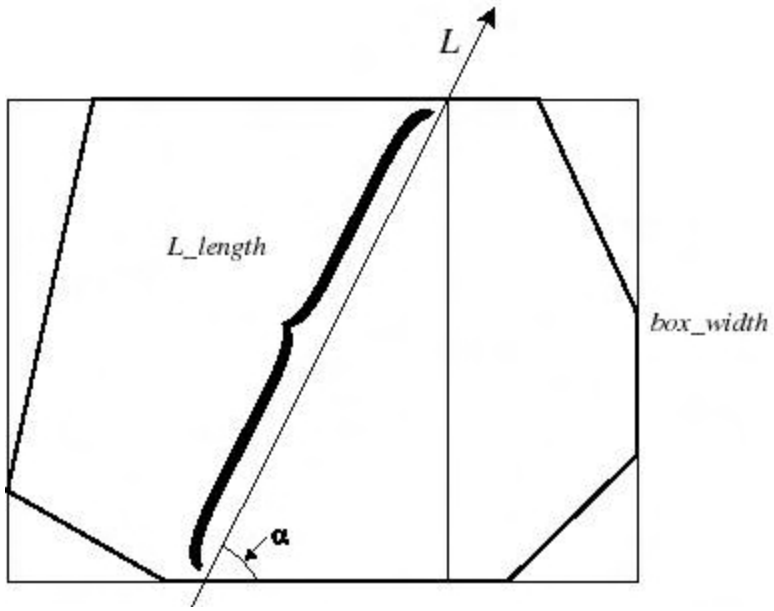


Figure 3.2 Triangular relation

To map the pixel length to ground truth vehicle lengths, we use the empirical vehicle length distribution shown in Figure 3.3 [15]. This allows us to readily obtain the ratio of the physical length, *L_physical*, and the pixel length, *L_pixel*, which is the scale factor *s*,

$$s = \frac{L\_physical}{L\_pixel} \quad (\text{ft / pixel}) . \tag{9}$$

This will play a key role in the next step.

3.3 DISTANCE AND SPEED ESTIMATION

Next, we estimate the travel distance between frames and the vehicle speed using the scale factor just obtained from the above geometric analysis.

First, some assumptions are made:

a) Distance traveled by a car is defined by the displacement of its centroid.

b) Scale change is smooth (linear with pixel distance) along the camera focus.

Therefore, all scale changes form an equal difference sequence.

c) Scale is homogeneous (constant) inside the car (box), so that the scale obtained from the ratio of the two lengths is equal to the scale factor at the centroid.

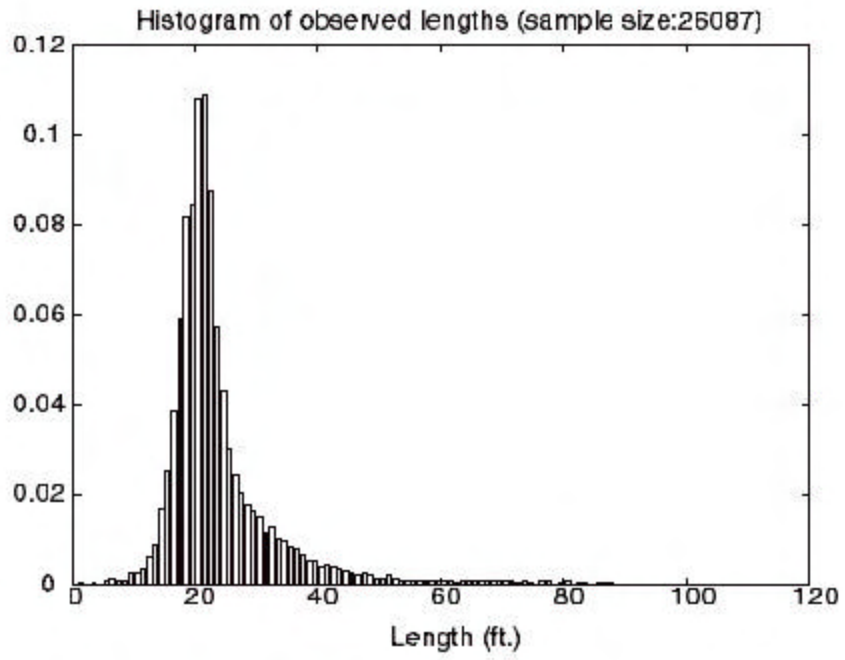


Figure 3.3 Vehicle length histogram

To obtain the physical distance of moving centers, we estimate the scale factors at each pixel along the travel path having angle α. To this end, we first need to compute the total number of pixels along the travel path, which can be obtained by using

$$\#\ of\ pixels\ along\ moving\ angle \approx \frac{\#\ of\ vertical\ pixels}{\sin\alpha}, \tag{10}$$

where the number of vertical pixels is simply the vertical pixel length between the first and the last centroids.

For an image sequence with k frames, where s_1 is the scale factor at the centroid of the convex hull of the vehicle of interest in the first frame, s_k is the scale factor at the centroid of the convex hull of the vehicle in the k-th frame, and the number of pixels along the driving path between these two centroids is n, we can compute the scale change per pixel, Δs, as

$$\Delta s = \frac{(s_k - s_1)}{(n-1)} \quad (\text{ft/pixel}^2). \tag{11}$$

The total distance D traversed between the images is then obtained by summing up the scale factor series as

$$\begin{aligned} D &= ns_1 + \frac{n(n-1)}{2}\Delta s \quad (\text{ft}) \\ &= n\frac{(s_1 + s_k)}{2} \end{aligned} \tag{12}$$

The speed is then estimated as the ratio of the interframe travel distance and the known frame rate.

The material just presented is the first published algorithm for an estimate of single vehicle speed using a statistical vehicle length and an uncalibrated camera. The algorithm creates scale information on the fly from information contained in the image and does not require calibration markers in the physical environment.

The algorithm presented here is validated against ground truth measurements in the next chapter.

4 Field Trials and Discussions

4.1 Field Trials Methodology

To test our algorithm, we compared the distance estimates (called estimated distance) obtained with our dynamic calibration technique with ground truth measurements on the freeway. Because the travel time interval is set by the inter-frame time, the only unknown is the ground truth travel distance. Field trials used both the distance between and the size of the center stripes. Both these measurements are published by WSDOT, as seen in Appendix A.

4.2 Experimental Results

Through extensive trials we tested the presented algorithm under different lighting conditions. Estimation error is defined as the difference of the ground truth distance and the estimated distance divided by the ground truth distance. Figure 4.1 shows the estimation error values and estimation error histogram for 60 image sequences. As suggested by Worrall et al [9], the mean car length, L_m, of 25.63 ft. is used in scale factor computations. The average estimation error for these 60 sequences is 8.7 percent.

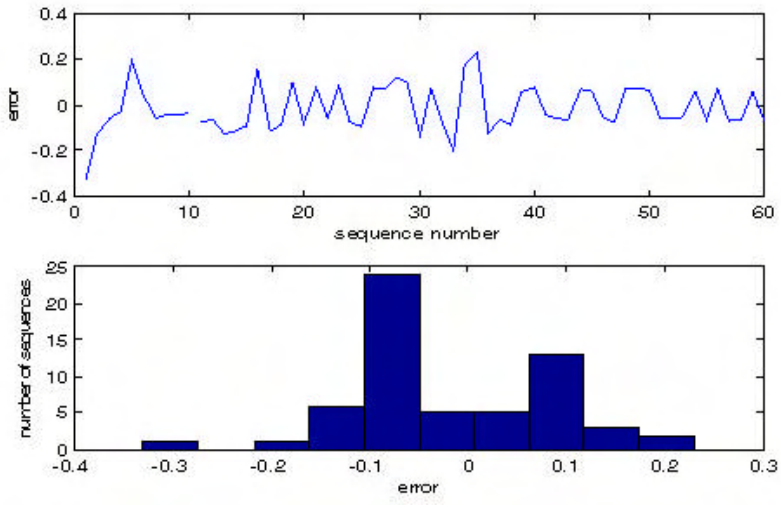


Figure 4.1 Errors and error histogram for 60 image sequences

Experiments suggested that when the area of shadow created by a vehicle is larger than two thirds of the vehicle area in the image, the estimation errors of our algorithm are unacceptable. We call this situation the severe shadow effect. Twenty such sequences were tested. The estimation error histogram is shown in Figure 4.2. Most produced estimation errors over 15 percent, some even over 25 percent. A typical image sequence with serious shadow effects is shown in Figure 4.3. Figure 4.4 shows two moving edge images for two frames in this sequence, where many moving edges actually represent the edges of shadows rather than those of the original vehicles. Initial analysis indicated that shadows will affect the reliability of the moving edge detection, the convex hull extraction, and, finally, the scaling computation, thereby distorting the distance and speed estimation. Without a priori knowledge of shadow shapes and directions, the effect of shadows cannot easily be included in this algorithm.

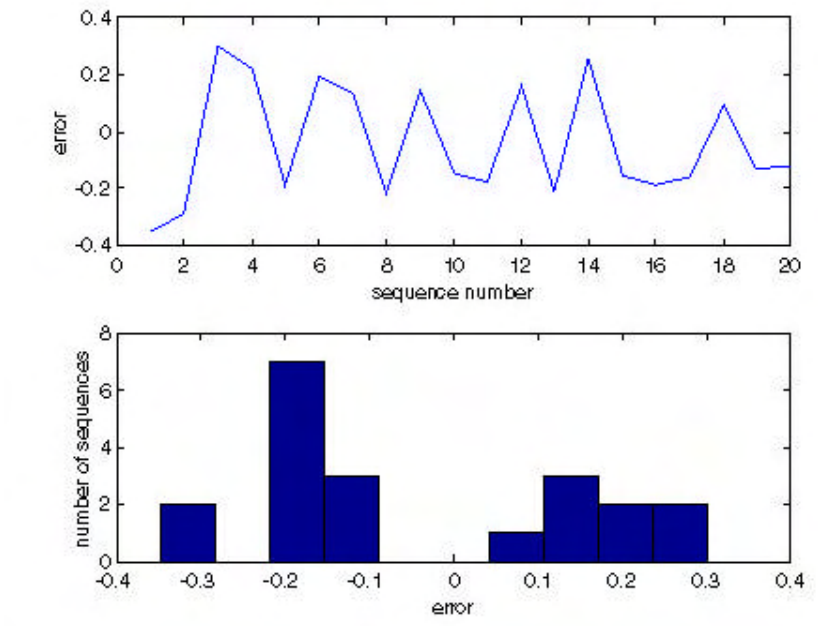


Figure 4.2 Errors and error histogram for 20 image sequences with severe shadow effects

Quantitative analysis of shadow effects is one focus for future improvements to the algorithm.

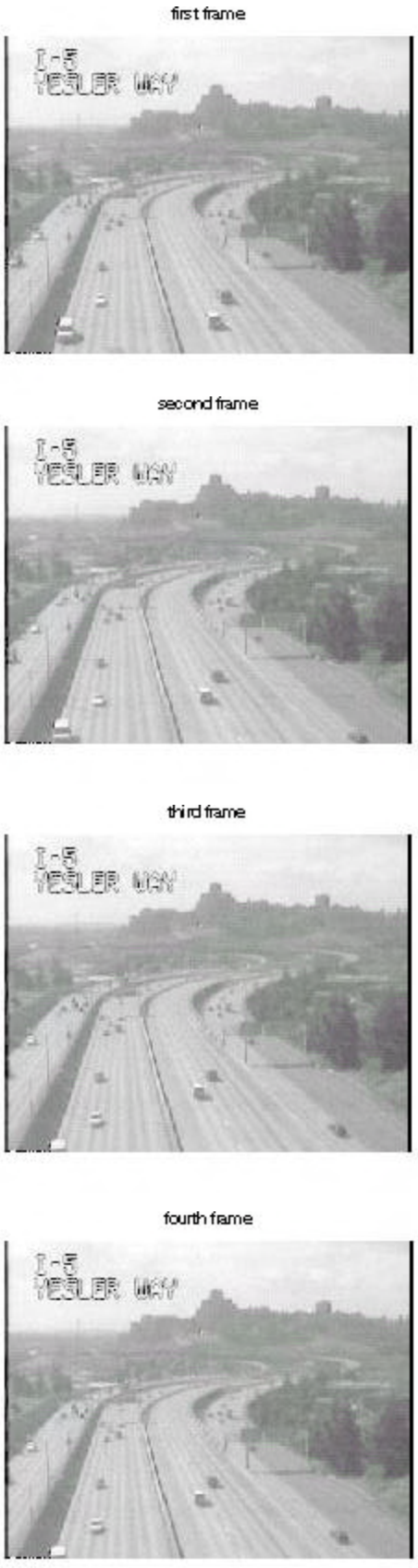


Figure 4.3 A typical sequence with severe shadow effects

Moving edges for the first frame

Moving edges for the fourth frame

Figure 4.4 Moving edges

4.3 ERROR ANALYSIS

From Worrall et al [9], the random variable vehicle length, *L_physical* (*L* is used below for simplicity), can be expressed as its expected value L_m (mean) and some deviation ΔL, that is

$$L = L_m + \Delta L \quad . \tag{13}$$

For an image sequence with k frames, suppose L_1 is the car pixel length in the first frame and L_k is the car pixel length in the k-th frame. Consider a case in which the cars are moving

away from the camera, so that the scale factor increases with the distance from the camera. (The analysis is similar for a case in which the cars are moving toward the camera.) Combining equations (9) and (12) gives us the estimated distance D_m,

$$D_m \approx \frac{n}{2}\left(\frac{L_m}{L_1} - \frac{L_m}{L_k}\right) \approx L_m \frac{n}{2}\left(\frac{1}{L_1} - \frac{1}{L_k}\right). \tag{14}$$

Considering length deviation in equation (13) gives us the deviated distance D_{de} as

$$D_{de} \approx (L_m - \Delta L)\frac{n}{2}\left(\frac{1}{L_k} - \frac{1}{L_1}\right). \tag{15}$$

Let e be the absolute error of distance measurement, and thus $e = D_{de} - D_m$. Combining equations (14) and (15) gives the mean of error e,

$$E\{e\} \approx \frac{n}{2}\left(\frac{1}{L_k} - \frac{1}{L_1}\right)E\{\Delta L\}, \tag{16}$$

where $E\{*\}$ is the expected value operator, and the variance, Var$\{e\}$, is

$$Var\{e\} \approx \left[\frac{n}{2}\left(\frac{1}{L_k} - \frac{1}{L_1}\right)\right]^2 Var\{\Delta L\}. \tag{17}$$

Equations (16) and (17) reveal that the length deviation (L) directly affects the measurement error, since the pixel number n and pixel lengths L_k and L_1 are uniquely determined for a specific image sequence with k frames.

4.4 POSSIBLE SYSTEM EXTENSIONS

The speed information obtained from this work can be used directly for many applications, such as traffic congestion detection. It is also worthwhile to note that with some modifications, our method can be readily extended to other traffic analysis, including incident

detection, traffic model verification, and travel time estimation. The techniques introduced in this report can be used as a basis for developing general-purpose, advanced intelligent traffic surveillance systems. For example, combined with character pattern recognition process, our method can be extended to recognize the vehicle license plate number, which has recently become an active research area.

5 Conclusion

There are many challenging problems in studying real traffic scenes within a complex background. In this report, efficient image processing techniques are applied to traffic analysis to estimate travel speed from image sequences of moving vehicles. Simple geometric relations are obtained directly from the image itself and are used to deal with real-world problems without explicit camera calibration. Furthermore, the techniques presented are validated against ground truth by field trials. Error analysis is also given in detail. The car length distribution is shown to be a key factor in the accuracy of speed sensing.

Some problems remain to be solved, including the effect of shadows and occlusion of vehicles.

As a result of the work presented here, a manuscript has been submitted to the IEEE Intelligent Transportation Systems Council for presentation at ITSC'99, a peer reviewed conference. A copy of this manuscript appears in Appendix B.

References

1. Kilger, M. Video-Based Traffic Monitoring. *International Conference on Image Processing and its Applications*, 7-9 April 1992, Maastricht, Netherlands, pp. 89-92.

2. Fathy, M. and M.Y. Siyal. An Image Detection Technique Based on Morphological Edge Detection and Background Differencing for Real-Time Traffic Analysis. *Pattern Recognition Letters*, Vol. 16, No. 12, December 1995, pp. 1321-1330.

3. Hoose, N. and L.G. Willumsen. Automatically Extracting Traffic Data From Videotape Using The CLIP4 Parallel Image Processor. *Pattern Recognition Letters*, Vol. 6, No. 3, August 1987, pp. 199-213.

4. Ali, A.T. and E.L. Dagless. Computer Vision for Automatic Road Traffic Analysis. *ICARCV 90, Proceedings of the International Conference on Automation, Robotics and Computer Vision*, 19-21 September 1990, pp. 875-879.

5. Fathy, M. and M.Y.Siyal, Real-Time Image Processing Approach to Measure Traffic Queue Parameters. *IEE Proceedings - Vision, Image and Signal Processing*, Vol.142, No.5, October 1995, pp. 297-303.

6. Soh, J., B.T. Chun, and M. Wang. Analysis of Road Sequences for Vehicle Counting. *1995 IEEE International Conference on Systems, Man and Cybernetics*, Vol.1, 22-25 October 1995, Vancouver, British Columbia, Canada, pp. 679-683.

7. Zifeng, J. Macro and Micro Freeway Automatic Incident Detection(Aid) Methods Based on Image Processing. *IEEE Conference on Intelligent Transportation Systems*, 9-12 November 1997, Boston, Massachusetts, USA, pp.344-349.

8. Picton, P.D. Tracking and Segmentation of Moving Objects in a Scene. *Third International Conference on Image Processing and its Applications (Conf. Publ. No.307)*, 18-20 July 1989, Warwick, UK, pp. 389-393.

9. Worrall, A.D., G.D. Sullivan, and K.D. Baker. A Simple, Intuitive Camera Calibration Tool for Natural Images. *Proceedings of the 5th British Machine Vision Conference*, 13-16 September 1994, York, UK, pp. 781-790.

10. Dickinson, K.W. and R.C. Waterfall. Video Image Processing for Monitoring Road Traffic. *IEE International Conference on Road Traffic Data Collection*, 5-7 December 1984, pp. 105-109.

11. Ashworth, R., D.G. Darkin, K.W. Dickinson, M.G. Hartley, C.L. Wan, and R.C. Waterfall. Applications of Video Image Processing for Traffic Control Systems. *Second International Conference on Road Traffic Control*, 14-18 April 1985, London, UK, pp. 119-122.

12. Takaba, S., M. Sakauchi, T. Kaneko, B. Won-Hwang, and T. Sekine. Measurement of Traffic Flow Using Real Time Processing of Moving Pictures. *32nd IEEE Vehicular Technology Conference*, 23-26 May 1982, San Diego, California, USA, pp. 488-494.

13. Hashimoto, N., Y. Kumagai, K. Sakai, K. Sugimoto, Y. Ito, K. Sawai, and K. Nishiyama. Development of an Image-Processing Traffic Flow Measuring System. *Sumitomo Electric Technical Review*, No. 25, January 1986, pp. 133-137.

14. Houkes, Z. Measurement of Speed and Time Headway of Motor Vehicles with Video Camera and Computer. *Proceedings of the 6th IFAC/IFIP Conference on Digital Computer Applications to Process Control*, 14-17 October 1980, Dusseldorf, West Germany, pp. 231-237.

15. Dailey, D.J. A Statistical Algorithm for Estimating Speed from Single Loop Volume and Occupancy Measurements. *Transportation Research B*, Vol. 33B, No. 5, June 1999, pp. 313-22.

16. Hoose, N. and L.G. Willumsen Real Time Vehicle Tracking Using the CLIP4 Parallel Processor. *Seminar on Information Technology in Traffic and Transport*. PTRC. Summer Annual Meeting, University of Bath, UK, 1987, pp. 113-131.

17. Koller, D., J. Weber, and J. Malik. *Robust Multiple Car Tracking with Occlusion Reasoning, Report No. UCB/CSD 93/780*, Computer Science Division (EECS) UC-Berkeley, 22 November 1993

18. Pratt, William K. *Digital Image Processing, 2nd Ed*. John Wiley and Sons. Inc., New York, c1991.

19. Davies, E.R. *Machine Vision: Theory, Algorithms, Practicalities*. 2nd Edition. Academic Press, London, England, 1997.

20. Ritter, Gerhard X. and Joseph N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra*. CRC Press, Boca Raton, Florida, 1996.

21. Gil, S., R. Milanese, and T. Pun. *Comparing Features for Target Tracking in Traffic Scenes. Pattern Recognition*, Vol. 29, No. 8, August 1996, pp. 1285-1296.

22. Kudo, Y., T. Yamahira, T. Tsurutani, and M. Naniwada. Traffic Flow Measurement System Using Image Processing. *33rd IEEE Vehicular Technology Conference*, Toronto, 25-27 May 1983, Ontario, Canada, pp. 28-34.

23. Cai, Q., A. Mitiche, and J.K. Aggarwal. Tracking Human Motion in an Indoor Environment. *International Conference on Image Processing*, 23-26 October 1995, Washington, D.C., USA, Vol. 1, pp. 215-218.

24. Vieren, C., F. Cabestaing, and J.G. Postaire. Catching Moving Objects with Snakes for Motion Tracking. *Pattern Recognition Letters*, Vol. 16, No. 7, July 1995, pp. 679-685.

25. Serra, Jean Paul. *Image Analysis and Mathematical Morphology*. Academic Press, London, New York, 1982.

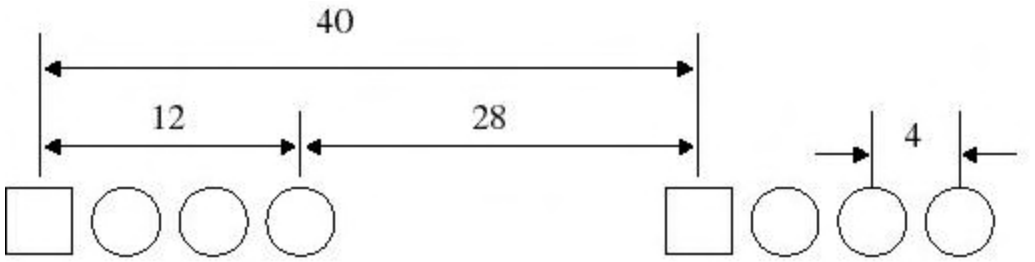
26. Fathy, M. and M.Y. Siyal. A Window-Based Edge Detection Technique for Measuring Road Traffic Parameters in Real-Time. *Real-Time Imaging*, Vol. 1, No. 4, October 1995, pp. 297-305.

27. Stewart, B.D., I.A.D. Reading, M.S. Thomson, C.L. Wan, and T.D. Binnie. Directing Attention for Traffic Scene Analysis. *Fifth International Conference on Image Processing and its Applications*, 4-6 July 1995, Edinburgh, UK, pp. 801-805.

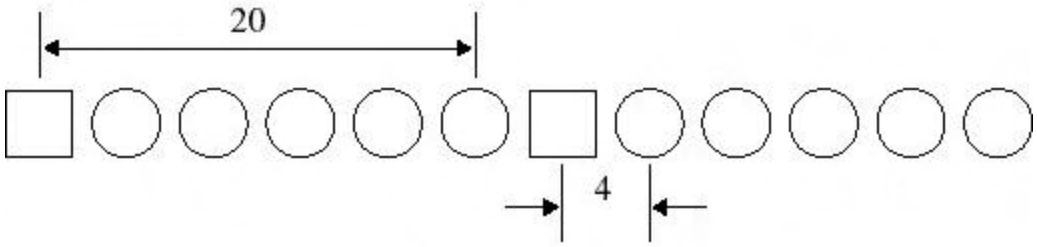
28. Bevington, Philip R. *Data Reduction and Error Analysis for the Physical Sciences*. McGraw-Hill Book Company, New York, 1969.

Appendix A

WSDOT Lane Stripe DATA



Skip Center Stripe



No Pass Stripe

Appendix B

Copy of manuscript submitted to IEEE Intelligent Transportation Systems Council for presentation at ITSC'99.